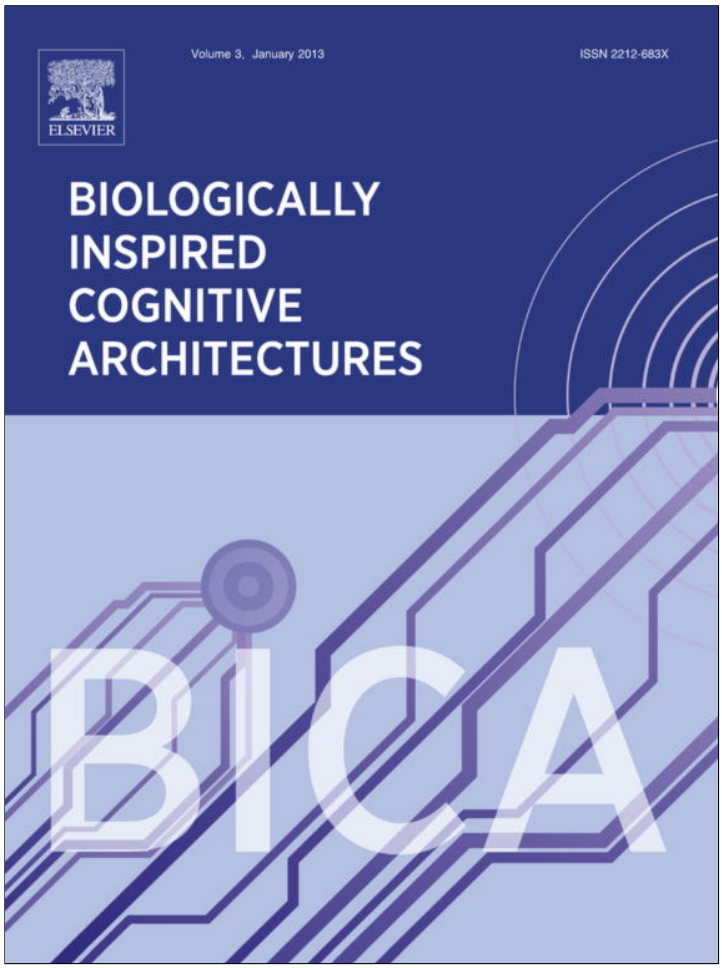


Biologically Inspired Cognitive Architectures (2013) 3, 87–104

Available at www.sciencedirect.com

SciVerse ScienceDirect

journal homepage: www.elsevier.com/locate/bica

RESEARCH ARTICLE

Distributed recurrent self-organization for tracking the state of non-stationary partially observable dynamical systems

Bassem Khouzam, Hervé Frezza-Buet *

Information, Multimodality and Signal, Supélec, UMI 2958 Georgia Tech/CNRS, Metz, France

Received 8 October 2012; received in revised form 31 October 2012; accepted 4 November 2012

KEYWORDS
Recurrent self-organization;
Partially observable dynamical systems;
Neural fields

Abstract
In this paper, a distributed recurrent self-organizing architecture is presented. It can extract the current state of a dynamical system from the sequence of the recent observations provided by this system, even if they are ambiguous. The recurrent network is an adaptation of `RecSOM` to the context of the simulation of large scale distributed neural architectures, since it relies on a strictly local fine-grained computation. The experiments show the ability of the recurrent architecture to capture the states, but also exhibit some unexpected dynamical effects, like some instabilities of the learned mappings. The presented architecture addresses the cognitive ability to set up representations from sequences at a mesoscopic level. At that intermediate level, between cognition and neurons simulation, some complex dynamics is unveiled. It needs to be identified and understood in order to bridge the gap between neuronal activities and high level cognition.

Introduction

Understanding how cognitive functions are produced by the human brain is a great challenge for neuroscience, which has implications in the field of public health of course, but also in the field of philosophy. In this paper, it is stressed that understanding the brain also offers new computational paradigms to computer science, for example in cognitive robotics in the context of situated artificial intelligence. Indeed, the brain is a complex dynamical system made of billions of neurons, exchanging electrical signals in parallel, in order to cope with the so-called brain–body-environment system (Beer, 2009). The information processing involved in the control of cognitive skills by the brain, such as learning and producing behavior, language, reasoning, recognition, dramatically differs from programs running on von Neuman machines. This processing is also far from solving explicit equations that model the system to be controlled,

* Corresponding author.
E-mail addresses: Bassem.Khouzam@supelec.fr (B. Khouzam), Herve.Frezza-Buet@supelec.fr (H. Frezza-Buet).

2212-683X/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.bica.2012.11.001

which is common in the automation domain. Moreover, trying to bring biological computational paradigms described by neuroscientists in the field of computer sciences is a way to test their efficiency in human-made information processing systems. Under this perspective, bio-inspired computer science also participates to the challenge of understanding the brain, bringing feasibility arguments to neuroscience ideas. The work reported in this paper proposes a contribution to such arguments.

The human brain consists of a set of sub-systems, each of them contributing harmoniously to the production of behavior, in spite of their specificity. This systemic view has been presented with a computational perspective in Doya (1999), for example. We have chosen to focus on the cerebral cortex, since its computational properties are very attractive for a computer science approach. Indeed, the cortical substrate seems to be able to cope with different modalities although the neural substrate is homogeneous, as suggested for example by Miller, Simons, and Pinto (2001) concerning the visual and somesthesic areas. Accordingly, the cortex is described as a bi-dimensional tiling of some replicated neural circuitry called a micro-column (Mountcastle, 1997), where regions responsible for some specific modality (visual areas, motor areas, auditory areas, somesthesic areas, etc.) have been found, as well as associative regions where different modalities are integrated (see Ballard, 1986; Burnod, 1989 for a global description). Today, the idea of a stereotypy of the micro-columns all over the cortex is still supported (Binzegger, Douglas, & Martin, 2005), but the debate is far from simple (Jones, 2000; Silberberg, Gupta, & Markram, 2002).

Our position is that the concept of cortical homogeneity is fruitful for computer science and that it is worth considering. Indeed, as the substrate is homogeneous, its specificity results from a self-organizing process that adapts autonomously the cortex so that it can handle some unexpected and not previously specified behaviors. Such reorganization of the cortical substrate, recruiting neurons related to obsolete representations for new functions, has been observed in amputated patients, or in the process of recovery after injuries (Elbert & Rockstroh, 2004). This recruitment of some neural space is very dynamical, since it is visible after only few hours (Stavrinou et al., 2007).

A self-organizing architecture is a dynamical process, but it can be used to represent static data, as for example the distribution of oriented contrasts in models of vision (von der Malsburg, 1973; Miikkulainen, Bednar, Choe, & Sirosh, 2005; Fellenz & Taylor, 2002). Addressing behavioral skills in the context of cognitive robotics rather involve dynamical representations. Indeed, behaving consists in scheduling actions, one after the other. This temporal description corresponds to a sequential decision making process, as addressed by reinforcement learning theory (Sigaud & Buffet, 2010). The corresponding brain structures involve sub-cortical systems, as the basal ganglia (Mink, 1996), but also the prefrontal cortex (see Fuster, 1997 for a global view, and Dominey, 1995 for early related models).

The present paper is thus an attempt to instantiate the ideas of homogeneous fine grained computation and self-organization in the context of the temporal processing of behavior.

Fine grained self-organizing dynamical systems as a basis for artificial cognition

The design of high performance computing resources for cortically-inspired systems is an emerging issue (Johansson & Lansner, 2007). The work presented in this paper participates to this challenge. Indeed, even if the results in this paper are obtained on few hundreds of computational units, it is implemented within the `InterCell` framework (Gustedt et al., 2011) that is dedicated to large fine grained dynamical systems, and thus enables further extensions toward larger scale systems. This perspective leads us to meet the following requirements in order to be compatible with the `InterCell` parallel framework, but also with the distributed nature of the neural computation in the brain:

- the system is strictly connectionist: it consists of units that update their state by *reading* the state of connected units and/or external inputs. The system is only defined by updating rules and connectivity. There are no global variables and no extra supervision of computation,
- the system is evaluated by successive updates of all the units, chosen in a random order, similarly to Hopfield networks (Hopfield, 1982),
- the system is only driven by its own dynamics and the external inputs. There is no separation between learning and test phases, nor reset when a new input is presented. Moreover, there is no need to feed in a signal indicating the start of a sequence. In other words, the system runs strictly online.

With such requirements, getting self-organizing properties within a population of units cannot be done by a winner-take-most (WTM) mechanism (Kohonen, 1997) since it requires the global computation of some best matching units. Indeed, WTM requires to synchronize the whole process by a global supervisor, in order to compute a global maximum, then to apply a learning strength kernel around it, and finally to learn according to that learning strength. Instead, for fitting the above requirements, some distributed competition within unit assemblies has to be set up, as detailed further.

Such competitive assemblies are called *maps* in the following and multi-map self-organization has been investigated in the past, leading to a general purpose architecture named `bijama` (Ménard & Frezza-Buet, 2005). This architecture, detailed further, actually instantiates the computational paradigms derived from the micro-column homogeneity hypothesis, and it is used here to address spatio-temporal self-organization.

Encoding time by recurrent architectures

As stressed previously, the production of behavior requires to take time into account, since behaving is a sequential decision process. In the field of artificial neural networks, recurrent connectivity is frequently used to provide the basis for the computation of dynamical features. Such a recurrent connectivity has actually been proposed for cortical models (Dominey, 1995) dealing with time. This way of

learning time dependent representations and processing within a dynamical context is thus in accordance with our cortical approach to computing.

A first class of recurrent neural networks is the class of feed-forward multi-layer perceptrons for which delay lines are added. These lines reinject the activities of some layer into the previous ones, with a delay of one time step. An overview of such networks, as well as the related gradient descent techniques (as for example Back-Propagation Through Time) for supervised learning of sequential data (time series, language) is proposed in Chalup and Blair (2003). The authors stress that the learning process is hard, and subject to a lack of generalization capabilities. They have to shape the network, by submitting more and more complex sequences to it, in order to drive the learning process.

Similar remarks about the learning complexity stands for a second class of recurrent architectures, made of neurons connected without any layered organization (Liu & Elhanany, 2007). In these architectures, the neural network is a dynamical system, driven by an updating rule (of the form $x_{t+1} = f(x_t)$, where x_t denotes the state of the system at time t) or by a differential equation $\dot{x}_t = f(x_t)$. Using an updating rule is suitable for reinforcement learning techniques, where synchronous interactions with the environment are used. This is addressed in Liu and Elhanany (2007) where temporal properties of recurrent networks are applied to the resolution of partially observable Markovian decision processes (POMDPs). In architectures based on differential equations, time is handled continuously, which complies with robotics requirements. For example, learning by imitation has been experimented on robots by continuous-time recurrent neural network (CTRNN) (Tani, Nishimoto, & Paine, 2008), implementing a collaboration between recurrent perceptivo-motor modules gated by a recurrent supervisor.

The problem related to these two classes of architectures, in the context of this paper, is that the learning techniques are based on complex gradient descent paradigms that do not fit the requirements expressed in Section 'Fine grained self-organizing dynamical systems as a basis for artificial cognition'. These techniques are rather motivated by some machine learning purpose and do not intend to fit any kind of biological plausibility. Nevertheless, let us mention works on CTRNN, closer to biology, that use evolutionary methods to design recurrent neural networks (Slocum, Downey, & Beer, 2000; Paine & Tani, 2004). The present work is not in the scope of such evolutionary techniques.

In order to face the learning problems encountered with the two previous classes of architectures, a third class of architectures based on reservoir computing approaches has been proposed. Such architectures are made of a *fixed* recurrent neural network (the reservoir) to which inputs are submitted, associated with an adaptive readout neural system. Under certain conditions, the reservoir is such that its instantaneous activity integrates the recent history of the system, and the readout system associates such instantaneous activity with some desired output. A review of reservoir techniques can be found in Lukoševičius and Jaeger (2009), and early models of thalamo-cortico-striatal loops in decision making assign the prefrontal cortex with a reservoir function (Dominey, 1995). Recent research addresses adaptive reservoir, implementing rules for intrinsic plasticity (Schrauwen, Wardermann, Verstraeten, Steil, & Stroobandt, 2008; Andreea Lazar & Triesch, 2009). The reservoir then adapts its dynamics to the statistics of the current input stream. An effort to allow for on-line computation, consistent with the requirements of Section 'Fine grained self-organizing dynamical systems as a basis for artificial cognition', has been made (Steil, 2007).

The idea underlying our approach is similar, in the sense that an adaptation of the recurrent part of the network driven by the inputs is proposed, but the focus is rather made on using topology-preserving neural networks for that purpose. This fourth class of architecture is detailed in the next section.

Fine grained recurrent self-organization

Sequence learning with cortically-inspired self-organizing architectures can be done by linking different units over the surface of some map according to their succession in the sequence. Such temporal connections are learned from specific spatio-temporal competition mechanisms (Durand & Alexandre, 1996; Wiemer, 2003; Frezza-Buet, Rougier, & Alexandre, 2001). A recent version of such mechanisms underlines the role of multi-winner competition processes can play in such computations (Schulz & Reggia, 2004). In the present paper, the approach rather consists in implicitly coding the sequence by the self-organization of a recurrent connection loop. Let us detail further this class of architectures.

Self-organization of the recurrent part of a temporal architecture, considering self-organization in the sense of Kohonen self-organizing maps (Kohonen, 1997), can be considered as giving a cortical flavor to the intrinsic plasticity of reservoirs presented previously. This is why it is of primary interest for our approach. An overview of recursive vector quantization techniques can be found in Barreto, Araújo, and Kremer (2003), Hammer, Micheli, Sperduti, and Strickert (2004) and a unifying formalism has also been proposed (Hammer, Micheli, Sperduti, & Strickert, 2004). Let us here summarize the process as follows. The ''reservoir'' is a self-organizing map, that receives inputs and arranges them on its surface according to some topology considerations. The point is that an input ξ_t received at time t is made of two components: the actual external observed input o^t and a context value c^t. The architecture is recurrent since c^t is computed from the state of the map at time $t-1$, when o^{t-1} was presented. Fig. 1 illustrates this work-flow.

This paper focuses on a distributed version of `RecSOM` introduced in Voegtlin (2002), that is a direct recursive version of the SOM algorithm by Kohonen (1997). We have indeed investigated distributed versions of SOM in previous studies (Alecu, Frezza-Buet, & Alexandre, 2011) and the mismatch of the winner-take-most part of SOM with our approach has been mentioned in Section 'Fine grained self-organizing dynamical systems as a basis for artificial cognition'. The present paper is an extension of this previous work to temporal computation. Let us mention here that stability issues are discussed in Voegtlin (2002), Peter Tiňo and Farkaš (2006). Stability is defined as follows in these works: the process with some constant input $o^t = o$, that is therefore reduced to some mapping from c^t to c^{t+1}, is considered stable when the mapping is a contraction (so

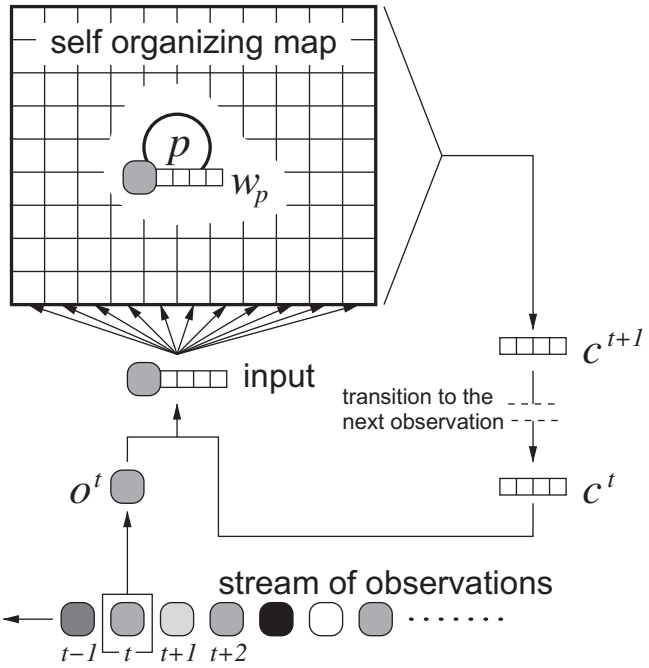


Fig. 1 Recursive self-organizing map principle. At each time, o^t is received from the input stream. It is combined with a context value c^t in order to form the actual input to all units within the self-organizing map. The map contains units p, each having a prototype w_p comparable to the input. The activities of units i are used to form c^{t+1} with a process that differs between different approaches (Hammer et al., 2004). c^{t+1} becomes the context value for the next input.

perturbations of c^t are damped). It is discussed in conclusion that other stability issues need to be considered in a distributed version of `RecSOM`, at the level of the self-organization process.

`RecSOM` has been applied to language models (Voegtlin, 2002; Farkaš & Crocker, 2008) but the present paper rather study the ability of recursive self-organization to retrieve some hidden state from observation. As our approach addresses the way the behavior of robotic agents can be produced by cortically-inspired architectures, an extension to POMDP is expected from the current work. This links our work with recursive architectures for POMDP as in Liu and Elhanany (2007) although the recurrent paradigm is not the same. Moreover, our architecture fits the `bijama` (Ménard & Frezza-Buet, 2005) model, as explained in Section 'Fine grained self-organizing dynamical systems as a basis for artificial cognition', which means that the recursive map presented here is expected to be included as an elementary component in a much larger neural system supported by some high performance computing middle-ware, where several of such recursive processes would be coupled.

Observation ambiguity resolving

In POMDP problems, the agent's environment is partially observable. The observations perceived by an agent may not allow it to determine the exact state of the environment. In spite of this, the agent, which receives a perceptive stream of observations and performs a sequence of actions as a response, has to take actions based on some estimation of the actual state of the environment. The problem occurs when identical observations correspond to distinct environment states. Knowing that different states could require different actions, the agent should resolve this ambiguity in order to decide the appropriate action to be taken.

The work presented here is more restrictive than POMDPs, since we consider an environment that changes at each time step according to some evolution function. Indeed, in POMDPs, the changes of the environment are triggered by the agent's actions. This will be addressed in further studies. So here, let us consider the environment as an autonomous dynamical system.

The actual state of the environment at time t is referred to as x^t. Its evolution through time is controlled by an evolution function ϕ_t, so that $x^{t+1} = \phi_t(x^t)$. The agent only observes the system's state and gets a stream of observations $o^t = O(x^t)$ through time (see Fig. 2a).

Let us consider the example of a wheel divided into gray-scale colored sectors (Fig. 2b). Each sector is labeled with a letter depending on its color. The wheel is turning counter clock-wise. We define the actual state x^t as the discrete angle of a fixed point on the disk circumference relative to an outer reference point. The observation is the color/letter at some fixed position (i.e. o^t in Fig. 2b). The sequence of observations is thus periodical, and it is actually *CDCEABEFCCFEDCBA* in the figure. In this example, observations are ambiguous. To illustrate this ambiguity, let us consider the color labeled *A*. In one wheel turn, this color could be observed in two different time instances $o^{t1} = o^{t2} = A$, corresponding to two different system states (wheel angle) $x^{t1} \neq x^{t2}$. In this case, if the agent is required to take different actions for each distinct state x^{t1} and x^{t2}, the observation *A* will not be sufficient to infer the system state.

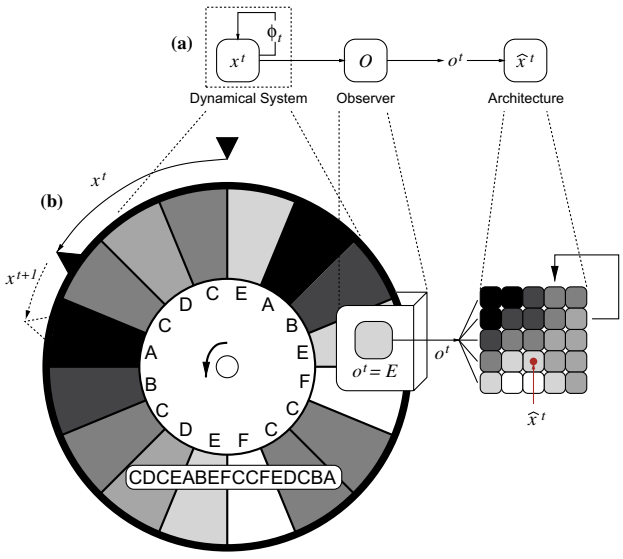


Fig. 2 Dynamical system state extraction. (a) A schematic of a dynamical system. (b) A toy dynamical system example that is a turning wheel exhibiting some observation ambiguity. The angle of rotation is the system state and the color of the angular sector at some fixed position is the observation. The observer gets identical observations for different states.

In this work, an unsupervised method processing on-line a stream of observations for setting up an internal state representation in proposed (see Fig. 2b right). This representation is expected to fit the actual states of the dynamical system in spite of ambiguity in the stream of observations. The proposed method uses a distributed fine-grained recurrent neural architecture based on self-organizing maps (SOMs), which is able, due to its recurrence, to consider the temporal context of the observations stream. In the case of ambiguous observations, as illustrated in the above wheel example, the architecture is expected to extract two different representations $\hat{x}^{t1} \neq \hat{x}^{t2}$ corresponding to $x^{t1} \neq x^{t2}$ although $o^{t1} = o^{t2}$. More precisely, a mapping of the state space over the map surface is expected, so that each state x is projected to some specific position $\hat{x}$ on the map. In further map representations, the observation o associated to each $x/\hat{x}$ is shown by the color assigned to the position $\hat{x}$ (see gray-scaled unit in the map on the right in Fig. 2b).

Last, let us stress here that the evolution function ϕ depends on time, as reminded in Fig. 2a. Indeed, in further experiments, the colors of the wheel sectors are changed during learning, i.e. the evolution function ϕ is suddenly modified. This allows us to test the ability of recurrent self-organizing process to cope with non-stationary dynamical systems.

Distributed recursive self-organization

Distributed winner-take-most

In Kohonen SOM, and in `RecSOM` as well (Voegtlin, 2002), the learning rate for prototypes is the highest for the unit whose prototype has a best matching to the current input. This learning rate is also non null for units in the surrounding (in the map) of the best matching unit. Such a distribution of the learning rates implements a winner-take-most competitive learning mechanism, dependent on the computation of the best matching unit, which is obtained from a global procedure that iterates over all the units. Indeed, once the best matching unit is found, a bell-shaped learning rate kernel (usually a Gaussian) centered on the best matching place is used to determine the learning rate of all the units. In order to fit our distributed paradigm, a fully distributed and dynamical winner-take-most mechanism has to be used. It is based on the neural field paradigm (Amari, 1977) that implements a distributed soft competition (i.e. a distributed winner-take-most). Using distributed competitive processes instead of a centralized winner-take-most in order to drive the self-organization reconciles our architecture with early models of cortical learning (von der Malsburg, 1973).

The competition is set up as follows. Each unit p in the map matches the input o^t it receives at time t against its internal weight values ω_p, thus producing a matching value distribution denoted i_p^t. The learning rates distribution $\left\{u_p^t\right\}_{p\in\text{map}}$ has to be obtained from a distributed dynamical process so that it implements the soft competition. This dynamical process computes the u_p^t so that they form a bump centered at the position of the highest i_p^t. As shown in Alecu et al. (2011), the neural field paradigm needs to be revisited in order to ensure self-organization under these conditions. The algorithm used in our experiment, named Lateral Input Sensitive neural field (`LISnf`) is derived from the neural field paradigm, but it has been adapted for a fast and reliable competition mechanism that suits a self-organization purpose. Thedynamical equations of `LISnf` are given in the appendix.

The architecture

In order to extract a representation of the states of the target dynamical system, we propose a multi-map fine-grained neural system. The system consists of three interconnected maps, one of which implements the time delay. It forms a recurrent pathway that captures the temporal history of the input stream, which in turn enables to overcome the observation ambiguity when extracting the state representation.

The architecture is built using the `bijama` model (Ménard & Frezza-Buet, 2005) developed in our team. `bijama` is well suited for dealing with cellular neural architectures and 2-D neural assemblies similar to the biological substrate of the cerebral cortex. Each map in the architecture contains an assembly of units. The three maps are the *input* map, *delay* map, and *associative* map (see Fig. 3). The activity of the units within the input and associative maps are computed by the `LISnf` neural field while the activities within the delay map are determined by the input map activity. The input map, which is the self-organizing one, receives at time t the external input o^t and builds a spatial coding $\hat{x}^t$ for it over its surface. The two other maps play an intermediate role in information running through the recurrent pathway. In particular, the delay map holds a delayed copy of the input map. The activities of both the input and the delay maps feed in the associative map. The activity of the associative map is re-injected in the input map. Thus, the input map dynamics will be updated on the basis of its external input in addition to the activity of the associative map. Consequently, an information about the past activity of the input which is held by the activity of the associative map intervenes in the input map dynamics. The evolution of the architecture parts is controlled by successive discrete time instances called *time steps*. At each time step, all the units in the architecture are updated once, following an asynchronous evaluation scheme, as mentioned in the appendix.

Information exchange within the architecture is carried through connections between its units. There are two types of connections, intra-map connections which implement a neural field (i.e. the on-and off-connections introduced in neural field oriented models of the cortical surface Amari, 1977), and inter-maps connections which form the recurrent pathway.

The intra-map connectivity allows the neural field to perform lateral competition. The field is parametrized in such a way that active units become agglomerated in one region of the map, i.e. the distribution of the activities takes the shape of a bump (see the dark meshes in Fig. 4b). The bump is actually the response of the map to its inputs whether they are external (for the input map) or internal (for the associative map) to the architecture. That is what `LISnf` actually computes. Activity bumps indicate the winning agglomerated set of neurons where learning should occur. Thus, lateral competition actually guides the process of

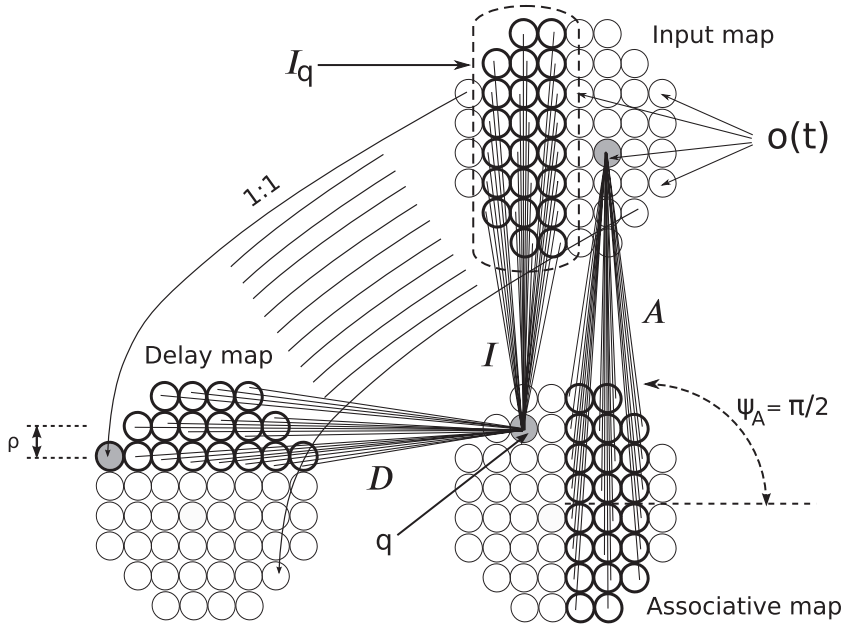


Fig. 3 Model architecture. Input map and delay map are connected via one-to-one connections, other connections are one-to-many connections organized in strips. For unit q,$\mathcal{I}_q$ is the strip of kind $\mathcal{I}$ handled by q.

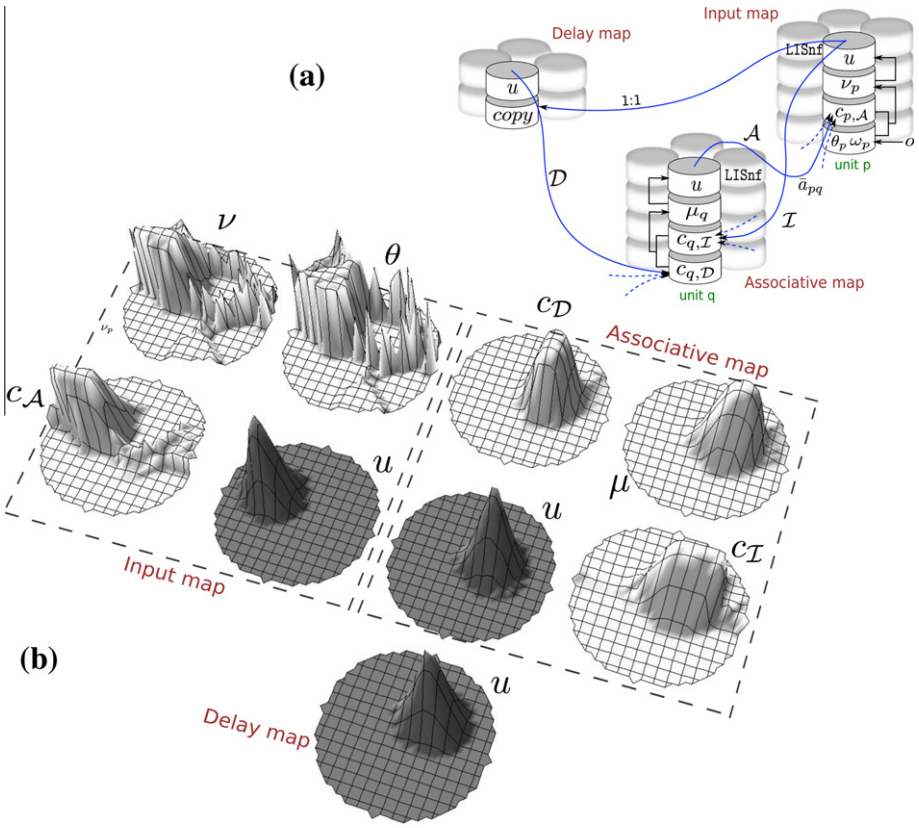


Fig. 4 A multi-map self-organizing architecture relying on the `bijama` framework. (a) Layered organization of the computation within the three maps. (b) Snapshot of all the activities in the architecture.

self-organization of the inputs over the self-organizing map. Once again, guiding self-organization using neural fields is revealed to be indeed difficult as explained in Alecu et al. (2011).

Inter-map connectivity means that each unit in some local map is connected to a set of units in a remote map. A unit at position p in the local map is connected to all the units of a strip-shaped region in the remote map (see

Fig. 3). This region is referred to as $\mathcal{S}_p$, thus, connections exist between p and the units at positions $q \in \mathcal{S}_p$. The set of strips handled by all local map units is denoted $\mathcal{S}$ (in Fig. 3 $\mathcal{S} = \mathcal{A}, \mathcal{I}, \mathcal{D}$, named after the map where the information originates: $\mathcal{A}$ for associative, $\mathcal{I}$ for input, and $\mathcal{D}$ for delay). These inter-map connections are referred to as *cortical connections*, reminiscent of neural connections between the different regions of the cerebral cortex (Scannell & Young, 1993). Each connection in the strip between the local unit p and the remote one $q \in \mathcal{S}_p$ has a weight denoted $\bar{s}^t_{pq}$ and referred to as *cortical weight*. The strip $\mathcal{S}_p$ owned by p handles a vector of cortical weights $\bar{s}^t_p = \left(\bar{s}^t_{pq}\right)_{q \in \mathcal{S}_p}$.

Strips are characterized by their width determined by the half width $\rho_{\mathcal{S}}$ and direction $\psi_{\mathcal{S}}$which is the angle between the horizontal axis and the axis connecting the centers of the local and remote maps (see Fig. 3). Let us denote u^t_p the activity of the unit at position p at time t and $s^t_p = \left(u^t_q\right)_{q \in \mathcal{S}_p}$ the vector of remote unit activities perceived at p through the strip $\mathcal{S}_p$.

In `bijama`, units activities are computed using a stack of modules (see Fig. 4a). All the units belonging to some map have the same stack composition. Each module in the stack handles a scalar value that is either received as an input or computed on the basis of scalars in lower modules. This leaves room to freely define the relation between module values respecting an ascending order dependency as explained later.

The uppermost module is the neural field module, which is actually the output u^t_p of the unit. This module is accessed by cortical connections for reading the unit activity. It implements the `LISnf` algorithm with input i being the scalar handled by the previous module in the unit's stack ($i = \mu$ for the associative map and $i = \nu$ for the input map).

Let us describe the composition of the stack of modules for each of the maps. See Fig. 4a and b while reading the descriptions that follows.

The input map receives the elements o^t of the stream of observations as external inputs. It also receives strips $\mathcal{A}$ from the associative map. Units of the input map compute their activities depending on these two inputs. The input map outputs an activity bump, whose position represents $\hat{x}^t$, as experiments show.

The lower module in the stack of a unit p in this map handles the external input o^t. This module is referred to as the *thalamic module*, reminiscent of the granular layer (IV) of the micro-column (Burnod, 1989). The thalamic module matches the input o^t against a stored weight ω^t_p called the *thalamic prototype*, and computes a similarity value θ^t_p, called the *thalamic matching*, decreasing with the distance between them:

$$\theta^t_p = \exp\left(-\frac{\left(o^t - \omega^t_p\right)^2}{2\sigma^2}\right) \tag{1}$$

Both the value of the thalamic prototype ω^t_p and the computed matching θ^t_p are stored in the thalamic module. The computed values of one module in all the unit stacks within the map form an activity distribution over the map relative to this module. Fig. 4b shows the activity of all modules in all maps. For instance, in Fig. 4b the computed matches θ^t_p for all the units of the input map form the thalamic matching distribution noted θ.

The second module in the stack for the input map is called the *cortical module*. It handles the strip $\mathcal{A}_p$ coming from the associative map. This module computes the matching $c^t_{p,\mathcal{A}}$ between the weight vector $\bar{a}^t_p$ of all the connections within the strip and the activity vector a^t_p of remote units in the associative map within the strip. The matching is computed as follows, where B is a numerical constant:

$$c^t_{p,\mathcal{A}} = \frac{\left\langle A^t_p . \bar{A}^t_p \right\rangle}{\max\left(\left\|\bar{A}^t_p\right\|^2, B\right)} \tag{2}$$

The third module computes a scalar that depends on both the thalamic and the cortical modules. It is called *cortico-thalamic merging* because it merges θ^t_p and $c^t_{p,\mathcal{A}}$ into one scalar v^t_p as follows:

$$v^t_p = \sqrt{\theta^t_p . \left(\beta + (1-\beta) . c^t_{p,\mathcal{A}}\right)} \tag{3}$$

where β is a constant that balances the participation of thalamic and cortical matches in their merge. Fig. 5 shows the variation of v^t_p as a function of θ^t_p and $c^t_{p,\mathcal{A}}$. The idea behind this rule is that the cortical activity is not sufficient to activate the unit. This principle can be found in Grossberg ART Model for visual attention (Grossberg, 1976). It stands that the input from other cortical regions is not sufficient to activate the neuron, but when a primary input exists it could modulate the neuron's activity. This is also compliant with the cortical sensibility in Burnod's model of the cortex (Burnod, 1989).

The value of v^t_p forms the input of the neural field, i.e. the upper module, which computes the unit activity u^t_p by executing some lateral competition between the units. This module is referred to as a *competition module*. Eq. (3) and Fig. 5 show that the map regions that have low thalamic matching θ will have low neural field inputs ν even if they have high cortical matching c. In other words, at each time step t, the activity u^t_p will be concentrated in regions where the external input o^t best matches the stored prototypes ω^t_p i.e. in regions having the higher θ^t_p. However, the participation of the cortical matching in the cortical merge is still important because it will determine the neural field bump position inside the region that has the best matching θ^t_p.

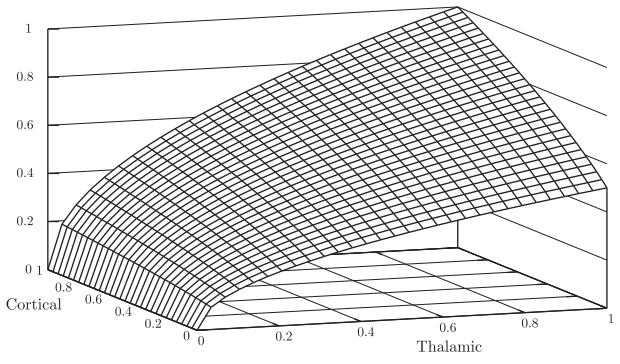


Fig. 5 Cortico-thalamic merge for $\beta = 0.25$.

In Fig. 4b, note that although the input to the neural field (noted ν) does not have the shape of a bump, the lateral competition within the neural field results in a bump of activity around the higher input values. Then, the unit activity value u_p^t computed by the neural field is used to modulate the learning of its thalamic prototype. Learning moves ω_p^t towards the input o^t proportionally to the unit's activity u_p^t:

$$\omega_p^{t+1} = \omega_p^t + \alpha_\omega . u_p^t . \left(o^t - \omega_p^t \right) \tag{4}$$

where α_ω is a fixed thalamic learning rate for all the units within the map. As u has the shape of a bump, only the units surrounding the best matching place actually learn.

Besides learning of the thalamic prototypes, the activities are used to modulate the learning of the cortical connections within the strips. For each connection, learning implies moving the cortical weight $\bar{a}_{pq}^t$ of a cortical connection towards the connection input u_q^t (activity of the unit in the remote map accessed by the connection).

$$\bar{a}_{pq}^{t+1} = \bar{a}_{pq}^t + \alpha_{\mathcal{S}} . u_p^t . \left(u_q^t - \bar{a}_{pq}^t \right) \tag{5}$$

where $\alpha_{\mathcal{S}}$ is a fixed cortical learning rate. Once again, the previous rule means that learning occurs only for the connections to active units in the local map. The learning mechanism of cortical weights in Eq. (5) is similar to that of thalamic weights in Eq. (4) responsible for the self-organization of thalamic weights in Kohonen maps (Kohonen et al., 2001). This means that the self-organization of cortical weights also occurs in cortical connections within a strip.

The second map in the architecture is the associative map: it receives the actual activity of the input map as well as its delayed activity, provided by the delay map. Both activities are conveyed by strips. This map performs lateral competition by means of the neural field to compute a bump shaped activity. Its activity is then read by the input map via the strips $\mathcal{A}$ as mentioned before.

The first module in the stack of a unit q in this map is a cortical module that handles the strip $\mathcal{D}_q$ from the delay map. The second module is a cortical module that handles the strip $\mathcal{I}_q$ from the input map. Both modules compute their matching $c_{q,\mathcal{D}}^t$ and $c_{q,\mathcal{I}}^t$ respectively, in the same way as explained in Eq. (2). The third module is referred to as *cortico-cortical merging*. It computes the merging μ_q^t of the previously computed cortical matching $c_{q,\mathcal{D}}^t$ and $c_{q,\mathcal{I}}^t$:

$$\mu_q^t = \sqrt{c_{q,\mathcal{I}}^t . c_{q,\mathcal{D}}^t} \tag{6}$$

The value of μ_q^t is actually the input to the associative map neural field module which computes the unit activity u_q^t. When u_q^t is computed it is then used to modulate the learning of the cortical weights within the strip $\mathcal{I}_q$ and $\mathcal{D}_q$ in the same way as in Eq. (5).

The last map in the architecture is the delay map, which receives a copy of the input map units activities via one-to-one connections and delays them for a specified period of T time steps. Its activity is injected in the associative map via the strips $\mathcal{D}$. The stack of a unit p in this map has two modules. The first is the *copy* module which copies its activity u_p^t from the activity of an input map unit u_q^t, where p is the position in the delay map, and q is the same position in the input map. The second module is the *FIFO* module which implements the delay. It contains a T-length FIFO queue that exposes its input on its output after T time steps. Thus $u_p^t = u_q^{t-T}$ for each unit p in the delay map and its corresponding unit q in the input map. The result is that the activities of the delay map at time t are the same as the activities of the input map at time $t - T$.

During the architecture presentation, it can be noticed that no prior knowledge about the dynamical system neither about the observation stream was required, thus it is a model-free architecture. After initialization, the parameters of the architecture are left unchanged even if the input stream changes. This is also true for the learning rates, which means that there is no need for progressively reducing the width of the soft competition kernel, as usually done in models based on self-organizing maps (Carpinteiro, 1999; Miikkulainen et al., 2005). This statement, in addition to the fact that there is no constraints on the initial conditions of the values of the thalamic and cortical weights, means that if the architecture succeeds in setting up a representation for a specific dynamical behavior, then it might succeed in re-adapting and setting up another representation when the system dynamics change.

Experiments

In this section the capability of the proposed recurrent self-organizing architecture to find a suitable representation for the states of a dynamical system is tested. The adaptation of the extracted representation to some change of the evolution function ϕ_t of the dynamical system is investigated as well. The `RecSOM` algorithm (Voegtlin, 2002) is also implemented for the sake of comparison with the proposed architecture.

The ability of both architectures to reveal the dynamical system state is rooted in two properties: the self-organization ensures that different representations are assigned to different input values on the map space, and the recurrence ensures that the temporal context is integrated to differentiate identical input values o^t (that correspond to distinct x^t).

The following experimental scheme starts by testing the capability of the distributed implementation of a SOM to exhibit self-organization. The ability of the proposed recurrent architecture to perform self-organization while considering the temporal context of the inputs is then tested. It is shown how the proposed architecture is able to build up a representation of a dynamical system state even though its inputs are ambiguous observations.

The behavior of the proposed architecture is then compared to that of the `RecSOM` (Voegtlin, 2002) recurrent architecture. The ability of both recurrent architectures to build up a correct representation although the dynamical system evolution function ϕ_t change is also investigated.

Notations and representations

Before showing the results, it is appropriate to introduce how observations are generally sampled from a dynamical system, the way they are introduced to the self-organizing architecture and how the behavior of that architecture is visualized. This applies to all the experiments presented in this paper.

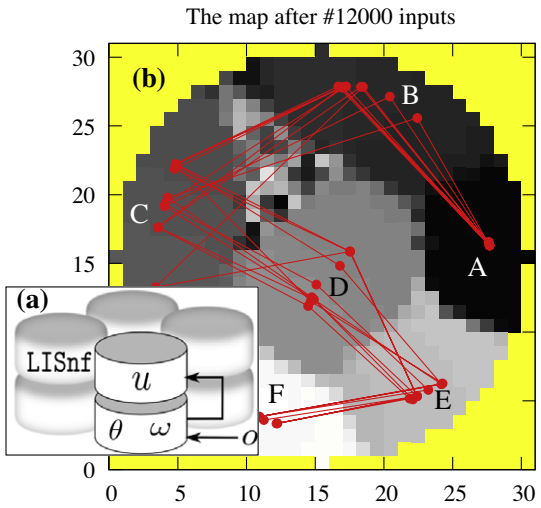


Fig. 6 (a) Module structure of an isolated input map. The module representation is the same as in Fig. 4a, and the equations of the modules are the ones detailed in Section 'The architecture'. (b) Distributed self-organization. The gray-scaled values correspond to prototypes ω_p^t, and the poly-line traces the succession of the $l = 50$ last bump positions in the map. Each point of the poly-line is labeled with the input presented at that time. The sequence here is *ABCDEFEDCB*.

The thalamic prototypes ω_p^t are coded in gray-scale colors (see Fig. 6b). Values close to 0 are in black and values close to 1 are in white. This color is used to represent the prototype of each unit p in the map, so that successful self-organization appears as a continuous shade over the surface. A region in the map where the thalamic prototypes are nearly uniform is called a *thalamic region* in the following. For example, there are two wide thalamic regions (black and white) in Fig. 12, while Fig. 7 (bottom-right) shows a map tiled with six ones.

The dynamics of the system is more difficult to illustrate. Let us denote x^τ the dynamical system state at a time instance τ, and $O(x^\tau)$ its observation. The observation is the external input to the input map (see Fig. 4a, on the right). The input $O(x^\tau)$ is maintained for several time steps in order to give sufficient time to the neural field to form the activity bump and to give enough time for the thalamic and cortical learning. It appeared experimentally that a small value of $T = 24$ time steps was enough for our experiments. This is due to `LISnf` that is quite fast. In previous work (Alecu et al., 2011), $T = 100$ was indeed required. In all the experiments, τ was incremented each T time steps, and the same input $O(x^\tau)$ was presented to the input map during the T time steps, so that: $o^t = o^{t+1} = \cdots o^{t+T-1} = O(x^\tau)$ and $o^{t+T} = o^{t+T+1} = \cdots o^{t+2T-1} = O(x^{\tau+1})$, etc., where o^t is the input for the input map at time t. The value of T is the same as the length of the FIFO queue of the delay map, i.e. the delay corresponds exactly to the duration T between two successive observations. So the time for the transitions in the dynamical system, that was denoted t on Fig. 2 is now denoted τ, since the letter t used for the time steps of the architecture evaluation.

As described in Section 'Observation ambiguity resolving', the input stream is the repetition of a periodic series of sampled observations $O(x^\tau)$, $O(x^{\tau+1})$, $\cdots$, $O(x^{\tau+n-1})$ where n is the period.

To give meaning to the response of the input map, the positions of its response (activity bumps) for successive input values $O(x^\tau), O(x^{\tau+1}), \cdots, O(x^{\tau+l-1})$ are plotted, where $l > n$. The goal of tracing l positions is to visualize the stability of the map response, this does not participate to any information processing in the architecture. The trace is obtained as follows.

The barycenter of the activity u of all map units is computed at the end of each T time steps, since at this time the neural field finds a stable bump for each input $O(x^\tau)$. The barycenter is computed as follows:

$$G^\tau = \sum_{p \in \text{input map}} u_p^t \cdot p \Big/ \sum_{p \in \text{input map}} u_p^t \tag{7}$$

The computed barycenter represents the map state $\hat{x}^\tau$ at time τ, i.e. the position of the activity bump, that is shown on the right in Fig. 2b as a red dot. From the stream of such $\hat{x}^\tau$ positions at each transition τ of the dynamical system, the trace of recent positions can be computed at each time τ. It is made of the last l barycenters ($l = 50$ in all experiments), organized in a list $P^\tau = \{G^{\tau-l+1}, G^{\tau-l+2}, \cdots, G^\tau\}$. They are drawn on a map representation in the form of a l-length path depicted as a red poly-line as in Fig. 6b and further figures.

All the experiments in this section are launched with the same numerical values.

For the dynamical system, the inputs are noisy. The value o^τ actually observed, and presented during T time steps, is indeed the value corresponding to the letter in the sequence: $A = 0$ (black), $B = 0.2$, $C = 0.4$, $D = 0.6$, $E = 0.8$ and $F = 1$ (white). This input value is added with a noise $noise_o$, but kept in $[0, 1]$ after the noise addition. This noise is sampled from a uniform random noise $\mathcal{U}[-0.05, 0.05]$. For experiments where the dynamics changes, the change occurs after the presentation of $\tau^s = 25{,}000$ inputs.

The parameters related to the architecture are the following ones. The map radius is $R = 15$ units for all maps, $u_p^0 = 0$, $\omega_p^0, \bar{a}_{pq}^0, \bar{i}_{pq}^0, \bar{d}_{pq}^0$ are initialized to uniform random values from $\mathcal{U}[0, 1], \sigma = 0.07$, $\alpha_\omega = \alpha_S = 0.00416$, $B = 10$, $\beta = 0.25$, $\rho_\mathcal{I} = \rho_\mathcal{A} = \rho_\mathcal{D} = 3$. $\psi_\mathcal{I} = 90°$, $\psi_\mathcal{A} = -90°$, $\psi_\mathcal{D} = 0, T = 24$, and $l = 50$. The competition module is implemented by the `LISnf` neural field equation detailed in appendix, with the following parameters: $\pi = 0.3$, $\delta = 0.333$, $\epsilon = 0.02$, and $\alpha = 20$.

For comparison, the algorithm `RecSOM` is used, and its parameters are set in order to fit the ones of our distributed architecture: $\lambda = 10$, $\eta = 0.1$, and $\Omega = 5$.

Self-organization in distributed winner-take-most methods

In this section, the architecture presented in Fig. 4a is reduced to the input map only, in order to fit to the classical self-organizing feature map (SOM) introduced by Kohonen (1997), except that the winner-take-most process is realized by some neural field. When a single input map is used, the two layers corresponding to the integration of the associative map influence are removed, so that the input map only consists of the thalamic module and the competition module on the top of it (see Fig. 6a).

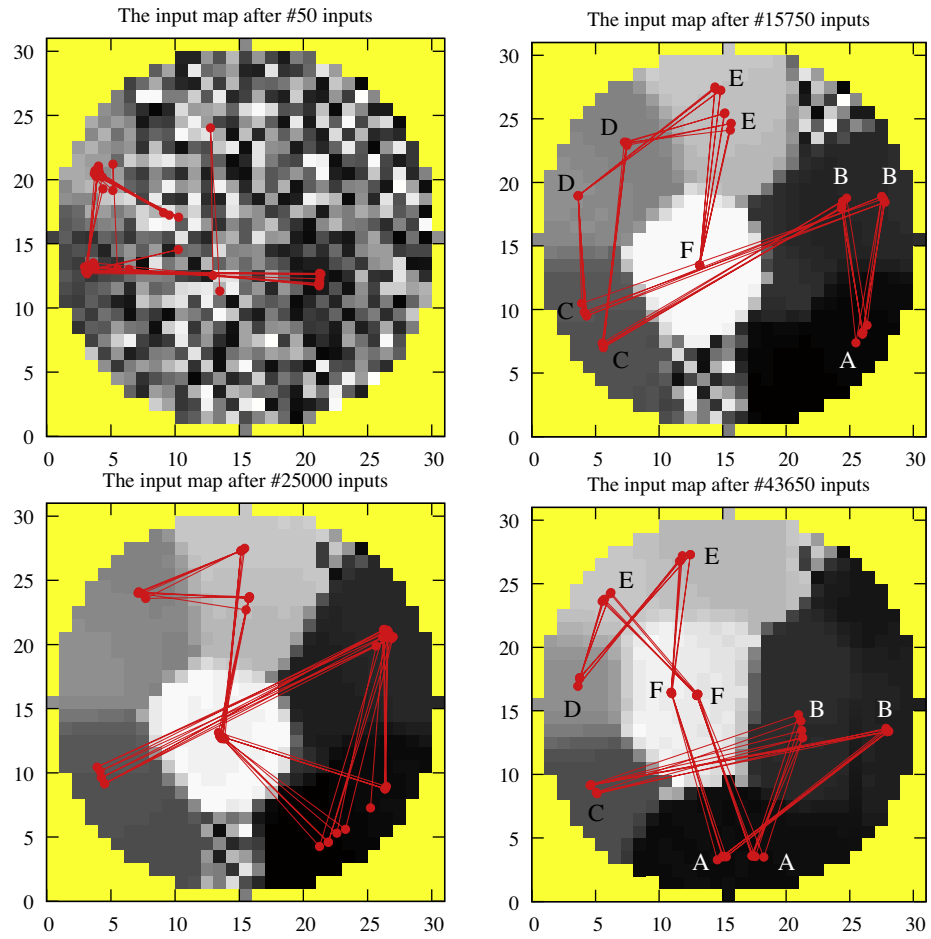


Fig. 7 Status of the input map during the system evolution as a response for both input series. The higher two figures represent the map response on the first input series S_1 = *ABCDEFEDCBA*. The two lower figures represent its response on the second input series S_2 = *ABCBAFEDEF*.

As one can expect, the input map behaves as a SOM, but this is not easy to obtain from usual neural field equations (Alecu et al., 2011). This is why the `LISnf` detailed in appendix is introduced here. The self-organization consists of a vector quantization of the input distribution, that preserves topology (neighboring colors are close within the map in Fig. 6b). The prototypes reflects that the input is taken from six values, from *A* to *F*, the noise being responsible for the spatial extension of the six regions in the map. The trace of the bump positions reveals that the input goes from *A* to *F*, and from *F* to *A*, but no distinction appears between the two directions, i.e the temporal context is not taken into account. For example, within the *C* region, there is no distinction between the observation of *C* received when the observation stream goes from *A* to *F* and the observation of *C* received when it goes from *F* to *A*.

Temporal disambiguation by recursive self-organization

In the former experiment, the SOM could not differentiate inputs that occur in different temporal contexts, and assigned inputs to clusters of points in the map depending only on the current input value.

The proposed architecture in this paper, depicted in Figs. 3 and 4, uses the principle of recurrence, thus involving the history of the input value in determining the corresponding representation in the map space. Recurrence lies in the use of multiple maps in the proposed architecture as seen in Section 'The architecture'. In this experiment, a repetition of the input series S_1 = *ABCDEFEDCB* is used, as in Section 'Self-organization in distributed winner-take-most methods', with the same noise. Let us recall that the weights within the strips as well as the thalamic prototypes are adapted continuously during the experiment.

Fig. 7 (top-left) shows the state of the input map when the experiment starts. The random initialization of the thalamic prototypes ω_p^t is still visible. The random poly-line shows that the map state is not stabilized at the experiment start. Fig. 7 (top-right) shows that the spatial self-organization of thalamic weights ω has occurred.

Each thalamic region corresponds to a range of observation values. Within each region there exists the representation of one or two states (see Fig. 7 (top-right)). Let us consider for example the black thalamic region, corresponding to observations close to 0. It contains the representation of one state marked *A*, this observation was not ambiguous. The two points marked *D* express non-successive states

corresponding to the same observation value, but considered in different temporal contexts. By following the polyline, it could be noticed that the order of *D* representations is conform with its order in the input series S_1, it is once preceded by *C* and once by *E*. The ambiguities of other values *B*, *C*, *E* are also resolved and two different state representations are assigned for each thalamic region. Thus, the architecture was able to resolve the ambiguities of the input values, and correctly assign them to different representations according to their input history. Unlike to Section 'Self-organization in distributed winner-take-most methods', the built up representations do not map only to observations values. Instead, they map one-to-one to the underlying states of the dynamical system so that the state representations in the map space $\hat{x}^\tau$ form a bijective mapping to the dynamical system state x^t.

This duplication of state representation corresponding to the same value of o^τ is progressively performed while the whole architecture gets organized, under the influence of the recurrent pathway that contains the delay. the strips bias the bump position within each thalamic region in the input map depending in the history of its activity. This allows to obtain multiple bump positions within the same thalamic region.

Tracking non-stationarity

The experiment presented in the previous section also aims at investigating the capability of the architecture to set up a new mapping of the dynamical system states when the system evolution function ϕ_t changes, i.e when the system is non-stationary. To test this feature, during the previous experiment, the sequence of input is changed, switching from the previous S_1 = *ABCDEFEDCB* to S_2 = *ABCBAFEDEF* at input $\tau = \tau^s$, with the same amount of noise.

As mentioned in Section 'Temporal disambiguation by recursive self-organization', some input values in S_1 are ambiguous, this is also the case for the values of S_2. The input value *A* in S_2 is preceded once by *F* and once by *B* so it is ambiguous, while *C* and *D* are not, as each of them is always preceded by the same series of values in S_2.

Fig. 7 (bottom-left) shows the input map state immediately after switching from S_1 to S_2. The past organization of the map which was fitting S_1 does not fit S_2. As an example, the input *F* which was corresponding to one representation in the map, is ambiguous now in S_2 and should be assigned to two distinct representations as it occurs in two different temporal contexts.

Fig. 7 (bottom-right) shows that the architecture reorganizes and finds another representation setup that forms a new correct mapping of the dynamical system states providing S_2.

Let us focus on two cases. The inputs *A* and *F* which were not ambiguous in S_1 are ambiguous in S_2 and the architecture assigns two distinct representations to these values in the new organization. For *A* and *F*, the corresponding regions on the map surface split when the sequence S_2 is presented. Contrarily, *C* and *D* which were ambiguous in S_1 and were assigned two distinct representations are not ambiguous anymore in S_2. Therefore they are each assigned only one representation in the new organization, i.e. the previously duplicated observation representation have merged.

Switching from S_1 to S_2 is done online. No reconfiguration of the architecture nor any parameter resetting is needed. As a result, the proposed architecture was not only able to set up a representation that maps to the state of the dynamical system, but it was also able to re-adapt online to track its non-stationarity, and set up a new correct representation when the transition function ϕ_t changes.

Comparison with the *RecSOM* model

In this section, The behavior of the proposed architecture is compared to that of RecSOM (Voegtlin, 2002) which is a recursive self-organization model. RecSOM combines SOM and feedback to represent input sequences.

The structure of RecSOM (Voegtlin, 2002) is similar to Fig. 1 so that the context information c^t is the map activity at the previous time step. Thus the inputs of some RecSOM unit *p* are the external input vector o^τ which is compared to a feed-forward weight vector ω_p^t, and the activity vector of all the units in the map at the previous time step which is compared to a weight vector $\bar{a}_p^\tau$. The activity of a unit *p* is a matching value computed as follows:

$$v_p^\tau = \exp\left(-\lambda\left\|o^\tau - \omega_p^\tau\right\|^2 - \eta\left\|a^{\tau-1} - \bar{a}_p^\tau\right\|^2\right) \tag{8}$$

where a^τ is the vector of all the activities $\left\{v_q^\tau\right\}_{q\in\text{map}}$ in the map.

In our experiments, the parameters λ and η are chosen so that their respective contributions in the exponential are of the same order, and so that the matching response v_p^τ has a sensitivity similar to what is obtained from Eqs. (1)–(3).

Like in Kohonen maps, the best matching unit is defined as the unit that minimizes v_q^τ. Let us denote $k^\tau = \text{argmin}_{p\in\text{map}} v_p^\tau$. Then, the learning rules used to update the feed-forward and the recurrent weights are:

$$\Delta\omega_p^\tau = \gamma h_{pk^\tau}\left(o^\tau - \omega_p^\tau\right) \tag{9}$$

$$\Delta\bar{a}_p^\tau = \gamma h_{pk^\tau}\left(a^{\tau-1} - \bar{a}_p^\tau\right) \tag{10}$$

where h_{pq} is a decreasing function of $d(p,q)$, the Euclidean distance between the positions of neurons *p* and *q*.

In Voegtlin (2002), Voegtlin used a very narrow neighborhood function *h* in the experiments, thus learning of feed-forward and recurrent weights occurs only for the winning unit. This implements a hard competition, i.e. a winner-take-all, rather than the soft competition (WTM) used here. This results in clustering inputs in K-means way rather than self-organization. In order to compare RecSOM to the proposed architecture, a slight modification of the neighborhood function used in Voegtlin (2002) is performed by replacing the exponential function h_{pk} by an arc of cosine, being non-null within a limited neighborhood of radius Ω. This fits the bump shape of the neural field competition process better than a Gaussian (see Fig. 4b, dark gray distributions).

$$h_{pk} = \begin{cases} \cos\left(d(p,k)\frac{\pi}{2}/\Omega\right) & \text{if } d(p,k) \leqslant \Omega \\ 0 & \text{if } d(p,k) > \Omega \end{cases} \tag{11}$$

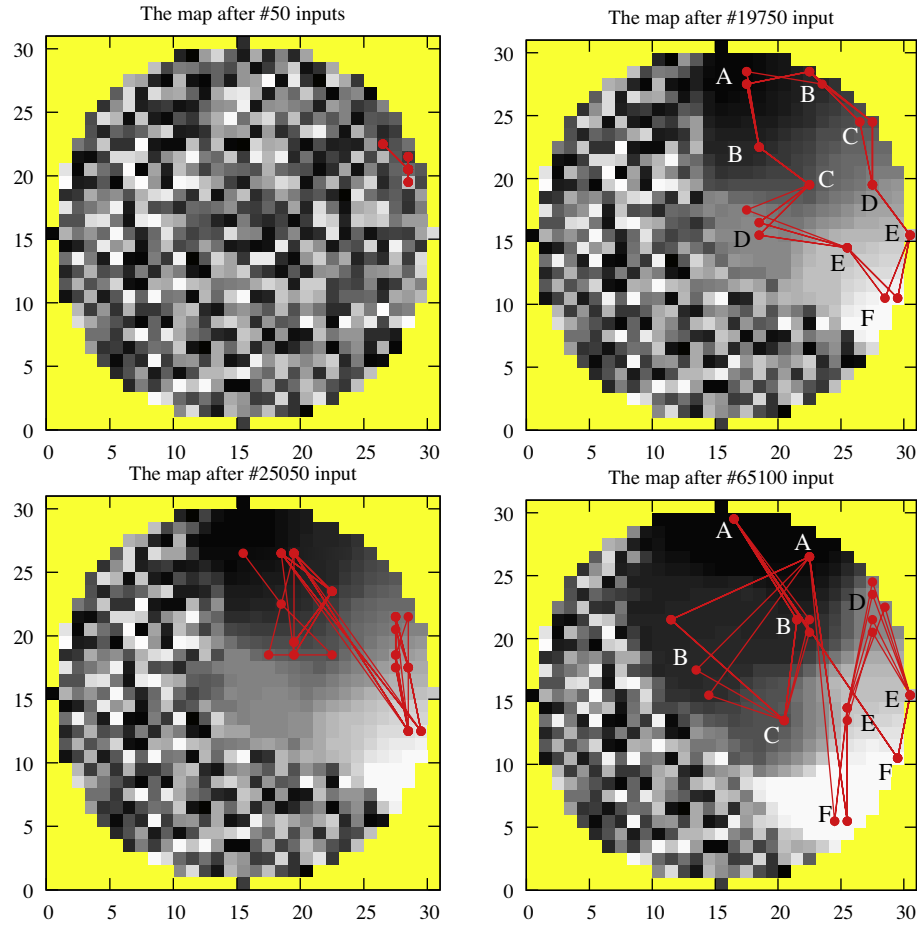


Fig. 8 Status of RecSOM during the system evolution as a response for both input series. The higher two figures represent the map response on the first input series $S_1 = ABCDEFEDCBA$. The two lower figures represent its response on the second input series $S_2 = ABCBAFEDEF$.

where $\Omega = 5$ is the bump width that is fixed to fit the width of the bumps observed in the proposed architecture. The RecSOM learning rate γ is chosen to be equal to the accumulation of the learning rates in the proposed architecture during T time step, i.e $\gamma = \alpha_\omega T$, since o^τ and $o^{\tau+1}$ are separated by T time steps in the case of our architecture.

Like in Sections 'Temporal disambiguation by recursive self-organization' and 'Tracking non-stationarity', RecSOM is tested with S_1 and S_2 with no parameter resetting, and with noise as well. Visualization issues in this experiment follows the same logic as before, except for one difference: a new input is introduced to the map at each time step, i.e $T = 1$, since the soft competition is not driven by a neural field that requires few time steps to relax to a steady state.

Fig. 8 shows the learned mapping using RecSOM. The initial state of RecSOM map is shown in Fig. 8 (top left). The feed-forward weight vector ω_p^t is initialized to random values as well. Fig. 8 (top right) shows the map state once the representation corresponding to S_1 is extracted. This is analog to Fig. 7 (top right) for our architecture. At this stage, RecSOM was able to set up a representation that resolves the observation ambiguity and map bijectively to underlying dynamical system states. By comparing Fig. 8 with Fig. 7, it could be noticed that all the map surface in the proposed architecture is recruited in the self-organization process, while in RecSOM only a part of the map was recruited.

Fig. 8 (bottom-left) shows the map state immediately after switching to the next input series S_2. Here, and similarly to Fig. 7 (bottom-left), the past organization which was fitting S_1 does not fit S_2 anymore.

Fig. 8 (bottom-right) shows how RecSOM was able to reorganize and find another representation setup that forms a new correct mapping to the dynamical system states corresponding to the observation series S_2.

As a result, like the proposed architecture, RecSOM was able to track the non-stationarity in the dynamical system, and set up a new correct representation when the dynamical system transition function ϕ_t changes.

Representation and stability issues

It has been shown in the previous section that our architecture is able to disambiguate observations provided by distinct states of the observed dynamical system, and that such recurrent self-organizing system can adapt to the changes in the dynamics of that system. The architecture

itself is a non-autonomous dynamical system (i.e. it is permanently influenced by the observation stream), that exhibits a complex dynamics since it is adaptive. Let us investigate in this section that complexity.

In the experiments described in the following, parameters are the same as previously, except that inputs are presented without noise (i.e. $noise_o = 0$).

Depth of the temporal context

The recursive architecture presented in this paper shares with RecSOM, but also with many other recurrent neural networks, the principle of a one time step delay ($\tau - 1$) re-entrance (see Fig. 1). Having a one step delay does not restrict the architecture to consider only the previous observation as a temporal context. For example, when the sequence of observations *AAAAAFF* is presented, the fifth *A* is characterized by the precedence of four *A*s before. With our architecture, initially, there is an emergence of only one region representing all *A*s, and one for all *F*s. Then, each of them split, creating new states, from which further splits are performed (see Fig. 9).

The RecSOM algorithm behaves similarly, but allows for deeper temporal context, as Fig. 10 shows. Moreover, it appears that the organization of prototypes by RecSOM is stable, while it continuously ''slides'' with our architecture. Indeed, in Fig. 11, different mappings of the same sequence can be observed at different time steps. This unstable mapping is discussed in the next section.

Mapping instabilities

During our experiments, some instabilities have been observed at the level of the mapping of the states of the external dynamical system over the input map surface. It means that the topology preserving mapping function, which is what the self-organization produces, continuously changes. Such mapping instability is by nature specific to topology preserving self-organizing systems, and it has not been reported in other studies of recurrent neural networks as far a we know. The instability issues addressed in RecSOM studies, mentioned in Section ''Fine grained recurrent self-organization'', are indeed a different problem.

Let us explicit this instability with a straightforward example. The system is provided with a short sequence of observations *AAFF*. It can be seen in Fig. 12 that the recurrent architecture proposed in this paper is able to map the four states of that sequence to four positions in the map surface in spite of observation ambiguity, as expected, but this correct mapping continuously drifts over the map surface (this also what happens in Fig. 11).

Sometimes, during the drift, some separate states merge inappropriately but re-split after a while. The same sequence is presented to RecSOM (see Fig. 13) that do not exhibit any drift.

In order to investigate the reasons for the occurrence of the mapping instability, let us consider again the input sequence *ABCDEFEDCB*, as in Fig. 7 (top right), but with smaller maps in our architecture. There is no input noise, as for

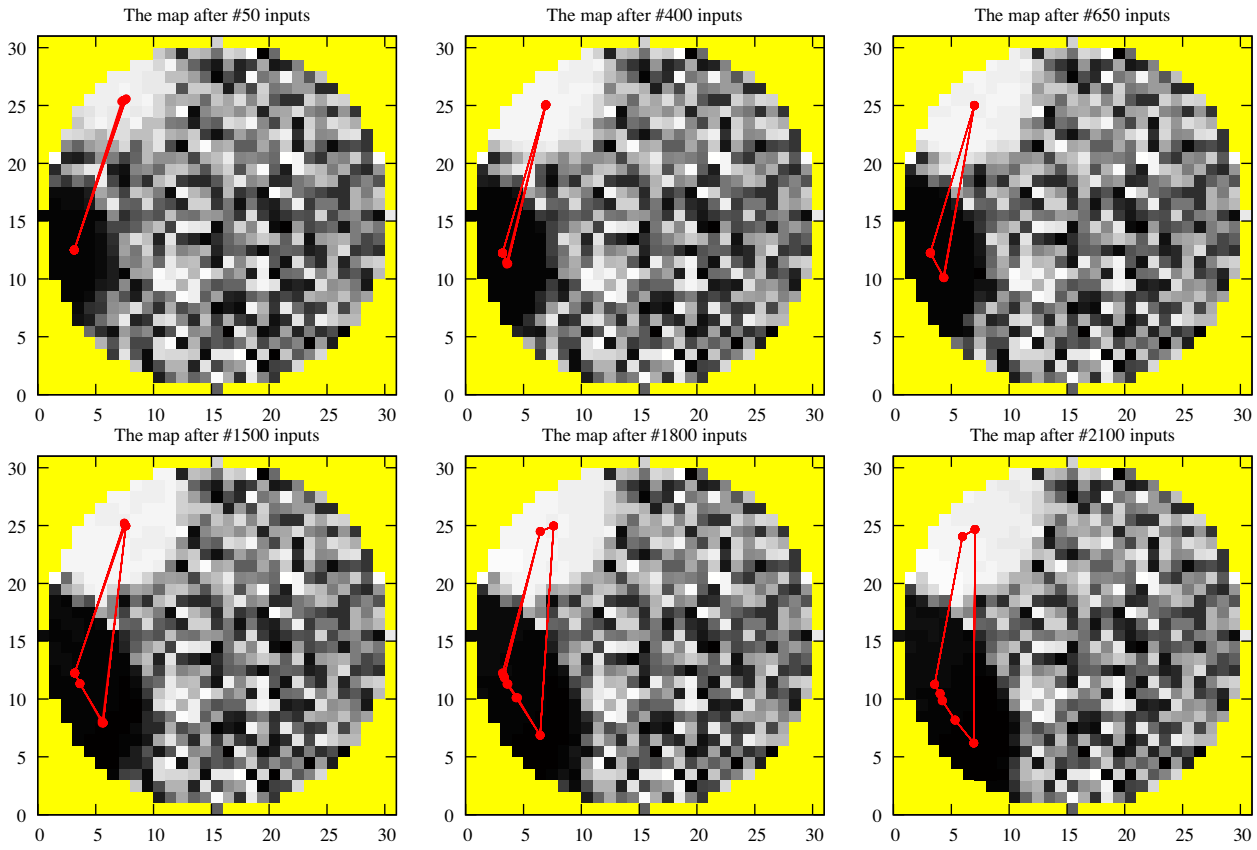


Fig. 9 Emergence of the mapping of the *AAAAFF* sequence.

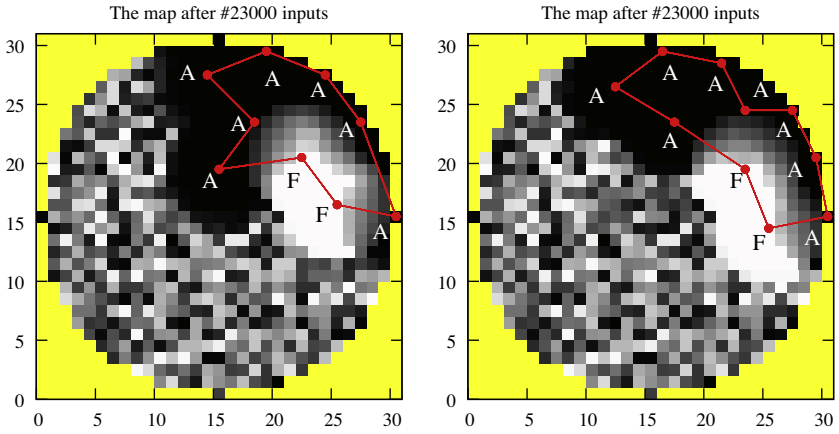


Fig. 10 `RecSOM` state after 23,000 inputs, for sequence *AAAAAAAFF* (left) and *AAAAAAAAFF* (right).

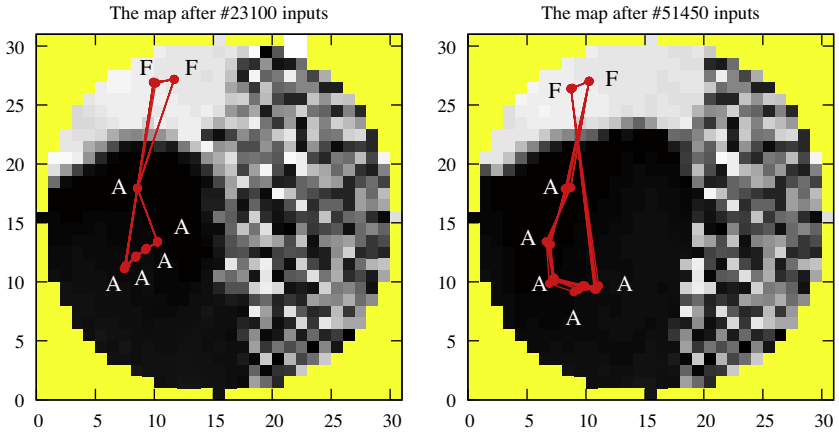


Fig. 11 Prototype organization by our architecture at different time steps. This is the continuation of the experiment in Fig. 9.

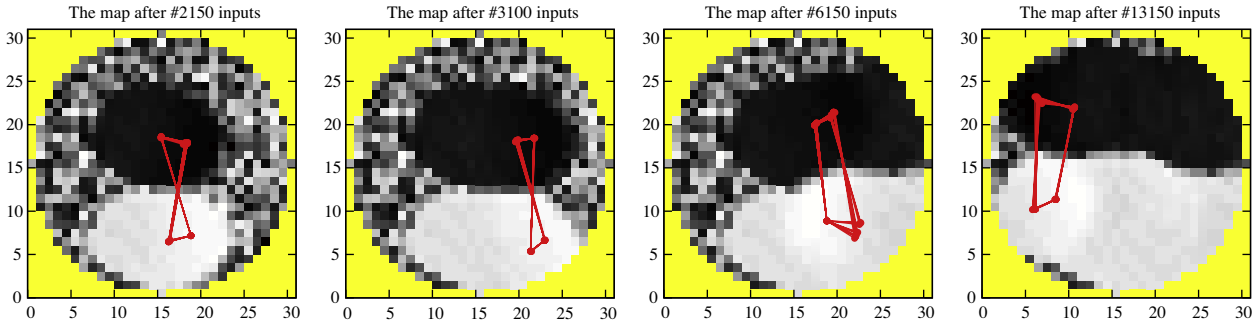


Fig. 12 A straightforward unstable mapping example. The input sequence is *AAFF*.

all other experiments reported in this section. The result is shown in Fig. 14a. The representation is stable, but the input map is not wide enough to allow for the splitting of *C* and *D* regions. When a simpler non-ambiguous sequence *ABCDEF* is presented to that small map architecture, the mapping instability appears (see Fig. 14b) as a smoothly rotating poly-line. Let us propose here a qualitative explanation of this difference. We have observed during the run of the experiments that the self-organization of the network looks like an expansive process: the vertices position of the poly-line evolve as if they were repulsive. This is visible in Fig. 9 where the distance between the nodes tends to increase when a region splits. For the sequence *ABCDEFEDCB* in Fig. 14a, the mapping of prototype is also in expansion, but it is kept confined within the small map so that expansion is limited by the lack of surface available for the representation. During the expansion, it seems that the vertices move in directions depending on the order in the input sequence. This may be due to the effect of the self-organization of weights within the strips between the maps. For the sequence *ABCDEFEDCB*, points *A* and *F* are subject to a change of direction along the poly-line. This may explain

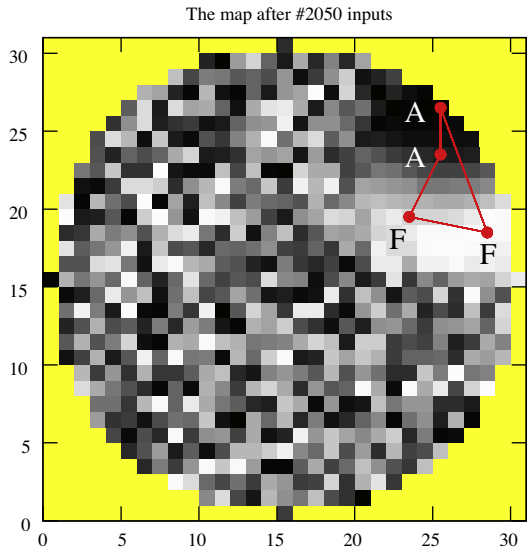


Fig. 13 `RecSOM` is stable for the *AAFF* sequence.

why point *A* and *F* in Fig. 14a are trying to move in opposite directions, which gives the stability in the confined space of the map. For sequence *ABCDEF* in Fig. 14b, there is no direction change and the vertices of the poly-line turn coherently with the sequence of inputs (counter-clockwise in our example).

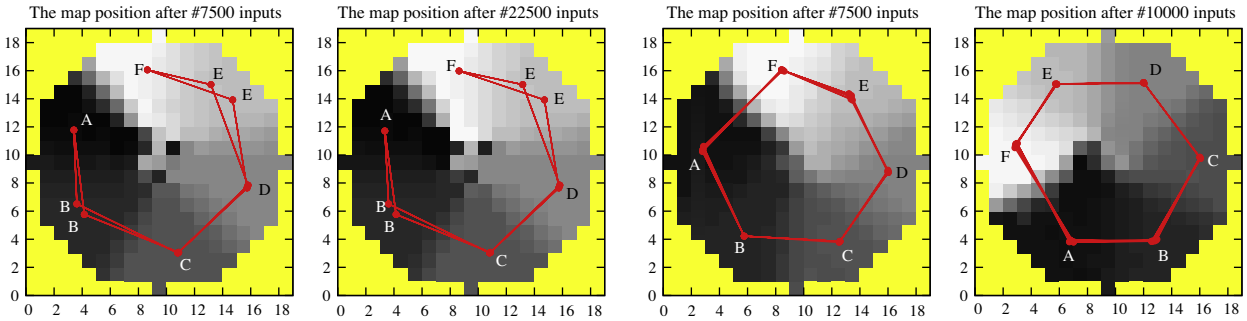
Unfortunately, this description is only qualitative, and a more formal approach of the mapping instability has to be done in further work. More precisely, the use of some limited and topographical connectivity (i.e. the strips) along the recurrent loop seems to influence the dynamics by adding mapping instabilities. Nevertheless, such connectivity has to be considered for wide architectures since all-to-all connectivity, as in `RecSOM`, grows exponentially with the number of units.

Discussion

In this paper, a distributed recurrent self-organizing architecture has been presented. Its design benefits from a more general framework, `bijama`, that allows for the definition of large multi-map neural systems. Indeed, the module organization as well as the strip connectivity help to build cortically-inspired systems that can be run on a cluster. The idea motivating `bijama` is to provide a construction set for complex, strictly distributed and multi-modal systems. Such computational requirements make this construction set suitable for large scale architectures, since it is internally designed for high performance computing. See Gustedt et al. (2011) for a justification, that is out of the scope of this paper. The principle of a self-organizing map driven by some neural field has been investigated in previous work (Alecu et al., 2011), as well as a generic connection scheme made of strips and associative maps (Ménard & Frezza-Buet, 2005). In this paper, the idea is to introduce and test a new kind of brick in that construction set, bringing into `bijama` the recurrent self-organization and the ability to cope with a temporal context.

As for the mentioned previous works where the Kohonen's self-organizing maps paradigm has been adapted to `bijama`, which are recalled by experiments in Fig. 6, the `RecSOM` algorithm is adapted here to a fine-grained distributed process. Nevertheless, bringing the `RecSOM` paradigm into `bijama` implies two major modifications. The first modification is the use of a neural field for the winner-take-most soft competition process, instead of applying a neighborhood function around the best matching unit. This point is not detailed in this paper, but it is critical (Alecu et al., 2011) and it is still at work. This explains why the `LISnf` algorithm is used for this purpose here, instead of more classical neural field equations. The second major modification is the temporal context itself (see c^t in Fig. 1). For `RecSOM`, the matching activity A^τ of the whole map is used as a context for all the units (see Eq. (10)). This matching distribution is analog to the ν_p distribution in the input map displayed in Fig. 4b, but with our architecture, a more reduced information is used as a context. Indeed, the *u* distribution (see Fig. 4b) is rather used, that is analog to the $h_{pk'}$ values in Eq. (10). Moreover, as opposed to `RecSOM`, the units do not share the same context, since they only view the remote *u* values from their limited connection strip. Such a connectivity allows us to define big maps, since it is less subject to combinatorial explosion than all-to-all recurrent connectivity of `RecSOM`.

The experiments have shown that the temporal self-organization shares the features of the more classical spatial self-organization. The architecture is indeed able to adapt to changes in the input distribution and it tracks the



(a) Sequence $ABCDEFEDCB$ (b) Sequence $ABCDEF$

Fig. 14 Mapping instability on a smaller map. See text for detail.

non-stationarity of the input, as Kohonen maps and RecSOM as well do. This leads to some forgetting of the previous distributions. The underlying view of the cortex that motivates our approach is actually oriented toward sensory-motor skills, even if actions are not considered for the moment. Under this perspective, changes in sensory-motor relationships rather correspond to long term modifications of the interaction with the world, and previous skills have to be forgotten.

Another result obtained from our experiments is the similarity between the behavior of our distributed architecture and RecSOM. Nevertheless, with our architecture, an instability is observed at the level of the mapping of the dynamical system states over the input map surface. With usual self-organizing maps, and RecSOM as well, several mappings of the inputs are actually possible, and the convergence reaches one of them. With our architecture, these possibilities seem to be continuously visited, by a smooth drift, except if the space of the map or the nature of the temporal sequence constraints that drift. This kind of instability has not been reported in other self-organizing systems, as far as we know, and it appears that addressing more complex computational systems unveils these complex dynamics. The representation of the dynamics introduced in this paper, i.e. the path of successive bump positions in the figure, enables to exhibit the effect, but more formal tools for analyzing what happens in the network are lacking.

Such a drift in the mapping have undesirable effects. For example, during the drift, correctly differentiated regions may collapse, and re-split afterwards. Moreover, as the architecture is proposed as a building block for more complex systems, unstable information may prevent the other blocks from being able to learn from bump positions within the self-organizing modules. In other words, it will be difficult to differentiate a change in the bump positions due to a drift from a change due to the non-stationarity of the dynamical system providing the inputs.

Forthcoming works will be twofold. First, a better understanding of the mapping instability will be investigated, from very simple examples as the one in Fig. 12. Second, coupling several recurrent self-organizing modules in bigger architectures has to be tested. This is feasible since bijama is designed for a cluster implementation. Such a test is relevant, even if the instability of the mapping is not solved, in order to see if this instability disappears when several modules are coupled. It is suggested by first experiments showed in Fig. 14 that having more constraints stabilizes the system.

Last, let us remind from Section ''Fine grained recurrent self-organization'' that our long-term goal is to design controllers in the field of cognitive robotics, the work presented in this paper is only a first step to introduce temporal representations in distributed self-organizing architectures. For this first step, the experiments are limited to the passive observation of some dynamical system producing ambiguous observations. An extension to POMDP, i.e. to the design of a controller from ambiguous observations, is more compliant with the cognitive robotics goal, but is not straightforward from our ''passive'' architecture, neither from RecSOM. This point needs further investigations.

The availability of high performance computing resources (Gustedt et al., 2011), as well as a methodology (bijama here) for the design of complex systems, opens the field of the simulation of complex computational systems, made of small non-linear units that interact massively. Such complexity is difficult to handle, due to emerging dynamics as self-organization as well as unexpected population effects, sometimes undesirable. Nevertheless, the nature of such complexity fits the nature of the information processing performed by our brains, and the difficulties that we have to face when we build such systems from scratch, from brain-inspired paradigms, may help to understand the relevance of the brain organization under the light of computational arguments.

Acknowledgement

This work is part of the InterCell Project supported by INRIA, Région Lorraine and Supélec. http://intercell.metz.supelec.fr

LISnf: Lateral input sensitive neural field

Let us note $\mathcal{D}_R(p)$ the units included in a disk of radius R around the unit p. Let us also note $|\mathcal{D}_R|$ its cardinal, i.e. the number of units in the disk. The connections that enables a unit p to read the activities of units $q \in \mathcal{D}_R(p)$ will be called on-connections. Let us define $\overline{\mathcal{D}}_R(p)$ as all the units in the map that do not belong to $\mathcal{D}_R(p)$. For a unit p, connections with units $q \in \overline{\mathcal{D}}_R(p)$ will be called off-connections. In order to save memory, a probability π for connecting unit p to some unit in $\overline{\mathcal{D}}_R(p)$ is used. The off-connections are thus rather noted $\overline{\mathcal{D}}^{\pi}_R(p)$. Last, let us mention that the computation of the LISnf dynamics requires to work on distribution $\bar{i}$ and $\bar{u}$ that are smoothed version of the input i and the activity u. This can be computed locally by averaging values over the on-connections.

$$\bar{i}^t_p = \frac{\sum_{q\in\mathcal{D}_R(p)} i^t_q}{|\mathcal{D}_R|},\ \bar{u}^t_p = \frac{\sum_{q\in\mathcal{D}_R(p)} u^t_q}{|\mathcal{D}_R|} \tag{A.1}$$

From these preliminary definitions, let us define an ordering relation between a pair $\left(\bar{i}^t_p, \bar{u}^t_p\right)$ at p and the analog values $\left(\bar{i}^t_q, \bar{u}^t_q\right)$ at q. Actually, a function $\mathrm{Inf}_t(p,q)$ returning 1 when the pair at q is greater than a pair at p is used. It is based on a lexicographic order with a priority of i over u, and on a tolerance factor $\epsilon \leqslant 0$:

$$\mathrm{Inf}_t(p,q) = \begin{cases} 1 & \text{if } \bar{i}^t_q > \bar{i}^t_p + \epsilon \\ 0 & \text{if } \bar{i}^t_q < \bar{i}^t_p - \epsilon \\ 1 & \text{if } \left|\bar{i}^t_q - \bar{i}^t_p\right| \leqslant \epsilon \text{ and } \bar{u}^t_q > \bar{u}^t_p + \epsilon \\ 0 & \text{otherwise} \end{cases} \tag{A.2}$$

The LISnf field equation is then a straightforward updating rule, that is not derived from any differential equation as with neural fields:

$$\begin{aligned} u^{t+1}_p &= \left[u^t_p + \delta \Delta u^t_p\right]^1_0, \\ \Delta u^t_p &= \left[1 - \alpha \frac{\sum_{q\in\overline{\mathcal{D}}^{\pi}_R(p)} \mathrm{Inf}_t(p,q)}{|\mathcal{D}_R|}\right]^{\pm} \end{aligned} \tag{A.3}$$

with $\alpha > 0$ and $\delta > 0$, $[x]^{\pm} = 1$ if $x \geqslant 0$ and -1 otherwise, and $[x]_0^1 = x$ if $0 \leqslant x \leqslant 1, [x]_0^1 = 0$ if $x \leqslant 0, [x]_0^1 = 1$ if $x \geqslant 1$. As mentioned in Section 'Fine grained self-organizing dynamical systems as a basis for artificial cognition', and accordingly to the `bijama` framework (Ménard & Frezza-Buet, 2005), Eq. (A.3) is applied to units in the map asynchronously. It means that, for each time step t, all the units are evaluated, according to some random order. There is no buffering, so an update of unit p is visible to any unit q whose evaluation occurs after p. This brings stability to the competition mechanism, that is fast and do not require any weights to be stored for on-and off-connections. A deeper study of the `LISnf` is in preparation, but is out of the scope of this paper.

References

Alecu, L., Frezza-Buet, H., & Alexandre, F. (2011). Can self-organization emerge through dynamic neural fields computation? *Connection Science, 23*(1), 1—31.

Amari, S. (1977). Dynamics of pattern formation in lateral-inhibition type neural fields. *Biological Cybernetics, 27*, 77—87.

Andreea Lazar, G. P., & Triesch, J. (2009). Sorn: A self-organizing recurrent neural network. Frontiers in Computational Neuroscience 3(23).

Ballard, D. H. (1986). Cortical connections and parallel processing: Structure and function. *Behavioral Brain Science, 9*, 67—129.

Barreto, G. deA., Araújo, A. F. R., & Kremer, S. C. (2003). A taxonomy for spatiotemporal connectionist networks revisited: The unsupervised case. *Neural Computation, 15*(6), 1255—1320.

Beer, R. D. (2009). The dynamics of brain-body-environment systems: A status report. In P. Calvo & A. Gomila (Eds.), *Handbook of cognitive science — An embodied approach. Perspectives on cognitive science* (pp. 99—120). Elsevier.

Binzegger, T., Douglas, R. J., & Martin, K. A. C. (2005). Cortical architecture. In M. De Gregorio, V. Di Maio, M. Frucci, & C. Musio (Eds.), *Brain, Vision, and Artificial Intelligence. LNCS* (Vol. 3704, pp. 15—28). Springer-Verlag.

Burnod, Y. (1989). *An adaptive neural network: The cerebral cortex*. Masson.

Carpinteiro, O. A. S. (1999). A hierarchical self-organizing map model for sequence recognition. *Neural Processing Letters*, 209—220.

Chalup, S. K., & Blair, A. D. (2003). Incremental training of first order recurrent neural networks to predict a context-sensitive language. *Neural Networks, 16*(7), 955—972.

Dominey, P. F. (1995). Complex sensory-motor sequence learning based on recurrent state representation and reinforcement learning. *Biological Cybernetics, 73*(3), 265—274.

Doya, K. (1999). What are the computations of the cerebellum, the basal ganglia and the cerebral cortex? *Neural Networks, 12*(7—8), 961—974.

Durand, S., & Alexandre, F. (1996). Tom, a new temporal neural net architecture for speech signal processing. In *Conference proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP-96)* (Vol. 6, pp. 3549—3552).

Elbert, T., & Rockstroh, B. (2004). Reorganization of human cerebral cortex: The range of changes following use and injury. *The Neuroscientist, 10*(2), 129—141.

Farkaš, I., & Crocker, M. W. (2008). Syntactic systematicity in sentence processing with a recurrent self-organizing network. *Neurocomputing, 71*(7-9), 1172—1179.

Fellenz, W. A., & Taylor, J. G. (2002). Establishing retinotopy by lateral-inhibition type homogeneous neural fields. *Neurocomputing, 48*, 313—322.

Frezza-Buet, H., Rougier, N., & Alexandre, F. (2001). Integration of Biologically Inspired Temporal Mechanisms into a Cortical Framework for Sequence Processing. In R. Sun & L. C. Giles (Eds.), *Sequence learning : paradigms, algorithms, and applications. Lecture notes in computer science* (Vol. 1828, pp. 321—348). Springer-Verlag.

Fuster, J. M. (1997). *The prefrontal cortex: Anatomy, physiology, and neuropsychology of the frontal lobe* (3rd ed.). Lippincott Williams and Wilkins Publishers.

Grossberg, S. (1976). Adaptive pattern classification and universal recoding: I. Parallel development and coding of neural feature detectors. *Biological Cybernetics, 23*, 121—134. http://dx.doi.org/10.1007/BF00344744, <http://dx.doi.org/10.1007/BF00344744>.

Gustedt, J., Vialle, S., Frezza-Buet, H., Sitou, D. B., Fressengeas, N., & Fix, J. (2011). Intercell: a software suite for rapid prototyping and parallel execution of fine grained applications. In *Applied parallel and scientific computing, 10th international conference, PARA 2010, proceedings, Part I*. In K. Jónasson (Ed.). *LNCS* (Vol. 7133). Heidelberg: Springer.

Hammer, B., Micheli, A., Sperduti, A., & Strickert, M. (2004). A general framework for unsupervised processing of structured data. *Neurocomputing, 57*, 3—35.

Hammer, B., Micheli, A., Sperduti, A., & Strickert, M. (2004). Recursive self-organizing network models. *Neural Networks, 17*(8-9), 1061—1085.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the USA, 79*(8), 2554—2558.

Johansson, C., & Lansner, A. (2007). Towards cortex sized artificial neural systems. *Neural Networks, 20*(1), 48—61.

Jones, E. G. (2000). Microcolumns in the cerebral cortex. *PNAS, 97*(10), 5019—5021.

Kohonen, T. (1997). *Self organizing maps* (2nd ed.). Springer.

Kohonen, T., Schroeder, M. R., & Huang, T. S. (Eds.). (2001). *Self-organizing maps* (3rd ed. Springer-Verlag, Inc.

Liu, Z., & Elhanany, I. (2007). A scalable model-free recurrent neural network framework for solving pomdps. In *IEEE international symposium on Approximate Dynamic Programming and Reinforcement Learning (ADPRL 2007)* (pp. 119—126).

Lukoševičius, M., & Jaeger, H. (2009). Reservoir computing approaches to recurrent neural network training. *Computer Science Review, 3*(3), 127—149.

Ménard, O., & Frezza-Buet, H. (2005). Model of multi-modal cortical processing: Coherent learning in self-organizing modules. *Neural Networks, 18*(5—6), 646—655 (extended version of *Coherent learning in cortical maps: A generic approach*, IJCNN'05).

Ménard, O., & Frezza-Buet, H. (2005). Model of multi-modal cortical processing: Coherent learning in self-organizing modules. *Neural Networks, 18*(5-6), 646—655, iJCNN 2005.

Miikkulainen, R., Bednar, J. A., Choe, Y., & Sirosh, J. (2005). *Computational maps in the visual cortex*. Springer.

Miller, K. D., Simons, D. J., & Pinto, D. J. (2001). Processing in layer 4 of the neocortical circuit: New insights from visual and somatosensory cortex. *Current Opinion in Neurobiology, 11*, 488—497.

Mink, J. W. (1996). The basal ganglia: Focused selection and inhibition of competing motor programs. *Progress in Neurobiology, 50*(4), 381—425.

Mountcastle, V. B. (1997). The columnar organization of the neocortex. *Brain, 120*, 701—722.

Paine, R. W., & Tani, J. (2004). Motor primitive and sequence self-organization in a hierarchical recurrent neural network. *Neural Networks, 17*(8-9), 1291—1309.

Peter Tiňo, J. v. M., & Farkaš, Igor (2006). Dynamics and topographic organization of recursive self-organizing maps. *Neural Computation, 18*(10), 2529—2567.

Scannell, J. W., & Young, P. (1993). The connectional organization of neural systems in the cat cerebral cortex. *Current Biology, 3*(4), 191—200.

Schrauwen, B., Wardermann, M., Verstraeten, D., Steil, J. J., & Stroobandt, D. (2008). Improving reservoirs using intrinsic plasticity. *Neurocomputin, 71*(7—9), 1159—1171.

Schulz, R., & Reggia, J. A. (2004). Temporally asymmetric learning supports sequence processing in multi-winner self-organizing maps. *Neural Computation, 16*, 535—561.

Sigaud, O., & Buffet, O. (Eds.). (2010). *Markov decision processes in artificial intelligence*. ISTE Ltd and John Wiley & Sons Inc.

Silberberg, G., Gupta, A., & Markram, H. (2002). Stereotypy in neocortical microcircuits. *Trends in Neurosciences, 25*(5), 227—230.

Slocum, A. C., Downey, D. C., & Beer, R. D. (2000). Further experiments in the evolution of minimally cognitive behavior: From perceiving affordances to selective attention. In: *Sixth International Conference on Simulation of Adaptive Behavior*.

Stavrinou, M. L., Penna, S. D., Pizzella, V., Torquati, K., Cianflone, F., Franciotti, R., et al (2007). Temporal dynamics of plastic changes in human primary somatosensory cortex after finger webbing. *Cerebral Cortex, 17*(9), 2134—2142.

Steil, J. J. (2007). Online reservoir adaptation by intrinsic plasticity for backpropagation decorrelation and echo state learning. *Neural Networks, 20*(3), 353—364.

Tani, J., Nishimoto, R., & Paine, R. W. (2008). Achieving organic compositionality through self-organization: reviews on brain-inspired robotics experiments. *Neural Networks, 21*(4), 584—603.

Voegtlin, T. (2002). Recursive self-organizing maps. *Neural Networks, 15*(8-9), 979—991.

von der Malsburg, C. (1973). Self-organization of orientation sensitive cells in the striate cortex. *Kybernetik, 14*, 85—100.

Wiemer, J. C. (2003). The time-organized map algorithm: Extending the self-organizing map to spatiotemporal signals. *Neural Computation, 15*, 1143—1171.